given from at first all being positive to mixtures of positive and negative and to finally all negative. In order to compare with Stevenson's approach and because it requires the bias to be zero, the biases of the six Adalines are all set to zero. In experiment, under the conditions that the elements of ΔW are all identical and the bias is zero for each Adaline, computer simulations that simulate Adalines' working process and computer computations according to the algorithm are separately run to compute the actual probability of erroneous outputs and the theoretical sensitivity for the six Adalines. Both the simulation results p and the theoretical results s given in Table I show that they are completely equal. This verifies the correctness of our approach. Further, the corresponding theoretical sensitivities s' based on Stevenson's approach for the six Adalines are also computed and listed in the last column of Table I. The comparison of the data in columns p, s, and s' demonstrates that our approach is more accurate.

VI. Conclusion

In this paper, a quantified sensitivity for Adalines to weight perturbation is given. The sensitivity is the basis of the study of Madalines' sensitivity. In addition, the sensitivity is hopefully expected to be a measure for evaluating importance of each Adaline in a Madaline, and so as to be helpful for training and pruning Madaline.

References

[1] M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of feedforward neural networks to weight errors," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 71–80, Jan. 1990.

[2] S. W. Piché, "The selection of weight accuracies for madalines," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 432–445, Mar. 1995.

[3] J. L. Bernier *et al.*, "Improving tolerance of MLP by minizing sensitivity to weight deviations," *Neurocomput.*, vol. 31, pp. 87–103, 2000.

[4] J. M. Zurada, A. Malinowski, and S. Usui, "Perturbation method for deleting redundant inputs of perception networks," *Neurocomput.*, vol. 14, pp. 177–193, 1997.

[5] A. P. Engelbrecht, "A new pruning heuristic based on variance analysis of sensitivity information," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1386–1399, Nov. 2001.

[6] X. Zeng and D. S. Yeung, "A quantified sensitivity measure for multilayer perceptron to input perturbation," *Neural Comput.*, vol. 15, no. 1, pp. 183–212, 2003.

[7] ——, "Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure," *Neurocomput.*, to be published.

Associative Memory Design for 256 Gray-Level Images Using a Multilayer Neural Network

Giovanni Costantini, Daniele Casali, and Renzo Perfetti

***Abstract*—A design procedure is presented for neural associative memories storing gray-scale images. It is an evolution of a previous work based on the decomposition of the image with 2^L gray levels into L binary patterns, stored in L uncoupled neural networks. In this letter, an L-layer neural network is proposed with both intralayer and interlayer connections. The connections between different layers introduce interactions among all the neurons, increasing the recall performance with respect to the uncoupled case. In particular, the proposed network can store images with the commonly used number of 256 gray levels instead of 16, as in the previous approach.**

***Index Terms*—Associative memories, brain-state-in-a-box (BSB) neural networks, gray-scale images, multilayer architectures.**

I. Introduction

The design of neural associative memories storing gray-scale images is a challenging problem investigated by few authors. Consider an image with n pixels and 2^L gray levels. The first approach is based on neural networks with multivalued stable states, a model introduced in [1]. The activation function is a quantization nonlinearity with 2^L *plateaus* corresponding to the gray levels. Denoting by n the number of pixels, the required number of neurons is n and the number of interconnections is n^2. Some design methods have been proposed for networks with this type of nonlinearity, with interesting experimental results [2].

A second approach is based on complex-valued neural networks [3]–[6]. The neuron state can assume one of 2^L complex values, equally spaced on the unit circle. Each phase angle corresponds to a gray level. The number of neurons is n; the number of interconnections is n^2. For complex-valued neural networks a generalized Hebb rule was proposed in [3], [4].

In [7], we proposed a third approach where each pixel can be represented by L bits, $b_1 \dots b_L$, so the image can be decomposed into L binary patterns with n components. Each binary pattern can be stored into a binary associative memory. There are L uncoupled networks, each with n^2 interconnections. The main advantage is that the L subnetworks can be implemented via parallel hardware with considerable saving in time, both for learning and recall. However, this approach presents two drawbacks. First, the storage probability of a random set of images is the product of the storage probabilities in the subnetworks. Hence, the capacity is quite lower than that of each subnetwork. In the same way, the recall probability, starting from noisy versions of the stored images, is reduced with respect to the recall probability of each subnetwork.

As the number of gray levels increases, both problems become worse since the number of independent networks increases. As a consequence, the method suggested in [7] is applicable only up to 16 gray levels. To overcome this limitation, we present an evolution of our previous approach based on the introduction of connections between layers. Building interlayer connections introduces interactions

Manuscript received October 8, 2004; revised May 26, 2005.

G. Costantini and D. Casali are with the Department of Electronic Engineering, University of Rome "Tor Vergata," Rome I-00100, Italy (e-mail: costantini@uniroma2.it; daniele.casali@uniroma2.it).

R. Perfetti is with the Department of Electronic Engineering, University of Perugia, Perugia I-06125, Italy (e-mail: perfetti@diei.unipg.it).

Digital Object Identifier 10.1109/TNN.2005.863465

among all the neurons; due to these interactions, both the capacity and the recall performance improve with respect to the uncoupled case. Taking into account the implementation issues, we limit the number of connections using an interconnecting structure much like that used in cellular neural networks (CNNs) [8], [9]. For this reason, in the following we borrow some of the notation from the CNN literature.

II. The Multilayer Neural Network

An image with 2^L gray levels can be decomposed into L binary patterns. Each pattern corresponds to a *layer* of the multilayer network. Each layer has $M \times N$ neurons (corresponding to the image pixels), arranged in M rows and N columns. Let (i, j, p) denote the neuron at the intersection of row i and column j in layer p ($i = 1, \ldots, M, j = 1, \ldots, N, p = 1, \ldots, L$). Each neuron is connected only to the neurons of its *neighborhood* defined by

$$N_{r,s}(i,j,p) = \left\{ (k,\ell,q) : |k-i| \le r \vee |k-i| \ge M-r \right. \\ |\ell - j| \le r \vee |\ell - j| \ge N - r \\ \left. |q-p| \le s \vee |q-p| \ge L - s \right\}$$

where r, s are positive integers; we assume $r \ge s$. Each neuron in layer p is connected to $(2r+1) \times (2r+1)$ neurons in each of the layers $p-s, \ldots, p \ldots, p+s$. To simplify the statements and the notation, *wraparound* connections have been assumed. For example, layer 1 is connected to layers $2, 3, \ldots, s+1$ and also to layers from L to $L-s$, so the number of connections per neuron is constant and equal to $\mu \doteq (2s+1)(2r+1)^2$. The proposed architecture consists of L layers, each with MN neurons; hence, the total number of neurons is MNL and the total number of interconnections is $\mu\ MN \log_2(2^L)$. The number of interconnections grows linearly with the number of pixels and logarithmically with the number of gray levels.

The state equation governing the time evolution of the network corresponds to the brain-state-in-a-box (BSB) neural model

$$x_{ijp}(t+1) = f\left[x_{ijp}(t) + \sum_{(k,\ell,q) \in N_{r,s}(i,j,p)} w_{ijp,k\ell q} x_{k\ell q}(t) \right] \quad (1)$$

where $x_{ijp} \in [-1, +1]$ is the state of neuron (i, j, p), $w_{ijp,k\ell q}$ denotes the weight of the connection from neuron (k, ℓ, q) to neuron (i, j, p), and $f()$ is the saturated linear function

$$f(y) = -1, \qquad y < -1$$
$$f(y) = y, \qquad -1 \le y \le +1$$
$$f(y) = +1, \qquad y > +1.$$

The decomposition of the image into L binary patterns was performed using the reflected-binary or gray code which has the property that, moving from a quantization level to an adjacent one, only one bit changes. Using the gray code instead of the usual binary-weighted code, additive zero-mean Gaussian noise on the gray-scale pattern results in a reduced Hamming distance between the stored binary pattern and the corresponding noisy pattern. As a consequence, the recall of a stored image is improved (see [7] for more details).

III. Learning Algorithm

To design the multilayer network, we adapt the algorithm proposed in [7], which is summarized for the reader's convenience.

We assume $w_{ijp,ijp} = 0$ for every i, j and p. Let $\underline{N}_{r,s}(i,j,p)$ denote the neighborhood of a neuron excluding the neuron itself. The design of the associative memory is as follows. First, we decompose the i-th gray-coded image to be stored into L binary patterns, from which we construct the i-th bipolar pattern $\mathbf{y}^{(i)} \in \{-1, +1\}^{M \times N \times L}$. Let $\mathbf{y}^{(1)} \ldots \mathbf{y}^{(Q)}$ be the Q bipolar patterns corresponding to the images to be stored. Then, we find the connection weights $w_{ijp,k\ell q}$ ($i, k = 1, \ldots, M$; $j, \ell = 1, \ldots, N$; $p, q = 1, \ldots, L$, $k, \ell, q \ne i, j, p$), satisfying the following set of constraints:

$$\sum_{(k,\ell,q) \in \underline{N}_r(i,j,p)} w_{ijp,k\ell q} y^{(m)}_{ijp} y^{(m)}_{k\ell q} \ge \delta > 0, \\ i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad p = 1, \ldots, L, \quad m = 1, \ldots, Q \quad (2)$$

where δ represents a stability margin.

To compute the weights, we use the following algorithm.

For every $n > 0$, compute

$$P\left(\Delta^{(m)}_{ijp}(n)\right), \\ i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad p = 1, \ldots, L, \quad m = 1, \ldots, Q$$

where

$$\Delta^{(m)}_{ijp}(n) = \sum_{(k,\ell,q) \in \underline{N}_{r,s}(i,j,p)} w_{ijp,k\ell q}(n) y^{(m)}_{ijp} y^{(m)}_{k\ell q} - \delta \quad (3)$$

and

$$P(x) = 0, \qquad \text{for } x \ge 0$$
$$P(x) = 1, \qquad \text{for } x < 0. \quad (4)$$

Then, update the weights as follows:

$$w_{ijp,k\ell q}(n+1) = w_{ijp,k\ell q}(n) + \eta \sum_{m=1}^{Q} y^{(m)}_{ijp} y^{(m)}_{k\ell q} \\ \times P\left(\Delta^{(m)}_{ijp}(n)\right) \eta > 0,\ (k,\ell,q \in \underline{N}_r(i,j,p)). \quad (5)$$

$P(x)$ is a penalty function of the constraint violation. η is a learning rate. Each term in the sum (5) can be $+1$, -1 or 0. As a consequence, starting from $w_{ijp,k\ell q}(0) = 0$, the weights can be represented as $w_{ijp,k\ell q} = \pm \eta N_{ijp,k\ell q}$, where $N_{ijp,k\ell q}$ is a positive integer. The required number of bits to represent the weights is $[\log_2(N_{\max})] + 1$, where $N_{\max}$ is the maximum value of $N_{ijp,k\ell q}$ and $[x]$ is the minimum integer greater than or equal to x. The properties of the algorithm are discussed in [10].

IV. Experimental Results

Two design examples are presented to show the effectiveness of the proposed neural network. In both examples, we assume 256 gray levels, ranging from 0 to 255 ($L = 8$), and $\eta = 0.1$ in (5).

Example 1: In this experiment, we investigate the noise removal capability. The design objective is to recall 50 stored images with 25×25 pixels. The images were randomly generated by the modified random midpoint displacement (RMD) algorithm proposed in [7], in order to approximate the histogram of real-life images, and stored using a network with $25 \times 25 \times 8 = 5000$ neurons and $r = s = 3$. We used two different values of δ, i.e., 100 and 500; correct storage of the 50 images is accomplished in both cases. Then, we tried to recall the stored images starting from corrupted versions. Noisy initial states were generated by adding zero mean Gaussian noise, with standard deviation σ, to each pixel of the stored images. In Fig. 1 the results are shown using $\delta = 100$ and $\delta = 500$, with a standard deviation σ from 10 to 35. The curves in Fig. 1 show the percentage of correct recall (no errors). As δ increases we observe an improvement of error correction. We repeated this experiment using the uncoupled neural network proposed in [7]. We stored the same images with 256 levels and we tried to recall them using the same values of σ (from 10 to 35); however, in this case the

TABLE I
PSNR VALUES IN dB FOR EXAMPLE 2

	noisy images ($\sigma = 29$)	recalled images with network in [7] ($\sigma = 29$)	noisy images ($\sigma = 19$)	recalled images with network in [7] ($\sigma = 19$)	recalled images with present network ($\sigma \leq 29$ or $\sigma \leq 19$)
Lenna	19.30	25.83	22.40	31.42	∞
Stefan	19.06	26.32	22.26	31.05	∞
Cups	-	-	22.67	27.96	∞

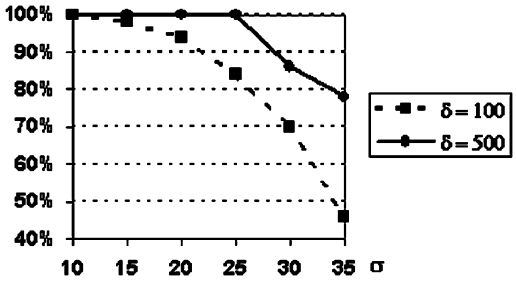

Fig. 1. Recall results of Example 1.

Fig. 2.Images Lenna, Stefan, and cups with 256 gray levels, used in Example 2.

percentage of correct recall is nearly zero. This example shows the poor performance achievable with the uncoupled solution when the number of gray levels is greater than 16.

Example 2: This design example concerns the real-life images with 200×200 pixels shown in Fig. 2; it is similar to the design examples used to test other methods [2], [6]. First, we store the images *Lenna* and *Stefan*. To sound better the capabilities of the proposed method, while keeping acceptable the computational cost, we partition each image into 16 parts, each with 50×50 pixels. In this way, we store 32 images using a network with $50 \times 50 \times 8$ neurons and 11 340 000 connections (assuming again $r = 4$, $s = 3$). Correct storage of the 32 subimages was obtained using $\delta = 800$. Then, we recalled the 32 stored subimages starting from noisy versions, generated by adding zero mean

Fig. 3. Noisy version of image Lenna with $\sigma = 29$.

Fig. 4. Recall results of the uncoupled neural network proposed in [7] storing Lenna and Stefan.

Gaussian noise with standard deviation σ. An example of noisy images with $\sigma = 29$ is shown in Fig. 3. All the images were correctly recalled without errors if $\sigma \leq 29$.

Then, we stored the three images with 200×200 pixels in Fig. 2 (*Lenna*, *Stefan*, *cups*) using the same method described above (with $r = 4$, $s = 3$). Therefore, we partition the three images obtaining 48 images, each with 50×50 pixels that we stored using a network with $50 \times 50 \times 8$ neurons and 11 340 000 connections. In this case, the three images are recalled without errors only if $\sigma \leq 19$.

Finally, we repeated this experiment with the uncoupled neural network proposed in [7], using the same images in Fig. 2. In this case, the network dynamics always converges to spurious images starting from noisy initial states. As an example, in Fig. 4 we show the best recall results for the *Lenna* image with two stored images (*Lenna* and *Stefan*) and $\sigma = 29$. Now we have considerable errors with respect to the stored images: These errors have the appearance of stains and stripes distributed all over the image. To evaluate the effect of noise, peak signal-to-noise ratio (PSNR) is very common in image processing. It is defined as

$$\mathrm{PSNR} = 20 \log_{10} \left(\frac{255}{\mathrm{rms}} \right)$$

where rms is the root mean square difference between the original image and the noisy version. The PSNR values for this design example are summarized in Table I. Using the multilayer neural network, we have $\mathrm{PSNR} = \infty$ (perfect recall), while using the method proposed in [7] we have an improvement of at most 9 dB.

V. Comparison With Existing Methods

In neural networks with complex neurons, the angle between adjacent complex points on the unit circle is inversely proportional to the number of gray levels. As a consequence, the recall performance deteriorates as the number of gray levels increases. For example, in [5] and [6] the number of gray levels used in the design examples are 8 and 20, respectively. The same problem occurs using multivalued activation functions with normalized output range (e.g., between -1 and $+1$ as in [2]). In the proposed method, the recall performance is satisfactory with the commonly used number of gray levels, i.e., 256.

A second aspect is the simple implementation. The learning algorithm described in Section III is completely local in the sense that the weights of each neuron can be computed independently from the others; so the learning algorithm is suitable to parallel computation. Moreover, it requires only add operations and a few bits of digital precision. The same advantages are shared by the network described in [7], which implementation is still more simple since the layers are uncoupled. However, the absence of coupling limits the number of gray levels, as explained previously.

The design method proposed in [5] requires complex arithmetics with a considerable amount of multiply and divide operations. The method proposed in [6] requires the solution of a linear programming problem as a design step; in order to have an acceptable computational complexity, the image must be decomposed into a large number of small subimages stored in independent networks.

Finally, we try to evaluate the *information capacity* of the proposed associative memory by comparing it to the classical Hopfield network with Hebbian learning. To this end, we estimate the storage capacity versus the *neighborhood size*, using synthetic images with 25×25 pixels; the images were randomly generated by the modified RMD algorithm proposed in [7]. We compute the ratio $\rho =$*number of equivalent binary patterns stored/number of interconnections*, in the case $r = s = 3$. We have 360×8 equivalent binary patterns stored with 100% probability; the number of connections is $343 \times 5000 = 1,715,000$. Hence, $\rho = 0.0017$. In the Hopfield network with 25×25 neurons, we have $\approx 0.15 \times 625 = 94$ stored binary patterns and $625^2 = 390,625$ connections, i.e., $\rho = 0.000\,24$. The present network gives an improvement factor of 7.

References

[1] J. M. Zurada, I. Cloete, and E. van der Poel, "Generalized hopfield networks for associative memories with multi-valued stable states," *Neurocomput.*, vol. 13, pp. 135–149, 1996.

[2] J. Si and A. N. Michel, "Analysis and synthesis of a class of discrete-time neural networks with multilevel threshold neurons," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 105–116, Jan. 1995.

[3] N. N. Aizenberg and I. N. Aizenberg, "CNN based on multi-valued neuron as a model of associative memory for grey-scale images," in *Proc. IEEE Int. Workshop Cellular Neural Networks Applications*, Munich, Germany, 1992, CNNA92, pp. 36–41.

[4] S. Jankowski, A. Lozowski, and J. M. Zurada, "Complex-valued multistate neural associative memory," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1491–1496, Nov. 1996.

[5] D. L. Lee, "Improving the capacity of complex-valued neural networks with a modified gradient descent learning rule," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 439–443, Mar. 2001.

[6] M. K. Muezzinoglu, C. Guzelis, and M. Zurada, "A new design method for the complex-valued multistate Hopfield associative memory," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 891–899, Jul. 2003.

[7] G. Costantini, D. Casali, and R. Perfetti, "Neural associative memory storing gray-coded gray-scale images," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 703–707, May 2003.

[8] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.

[9] ——, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1273–1290, Oct. 1988.

[10] R. Perfetti and G. Costantini, "Multiplierless digital learning algorithm for cellular neural networks," *IEEE Trans. Circuits Systems I*, vol. 48, no. 5, pp. 630–635, May 2001.

Convergence of Gradient Method With Momentum for Two-Layer Feedforward Neural Networks

Naimin Zhang, Wei Wu, and Gaofeng Zheng

***Abstract*—A gradient method with momentum for two-layer feedforward neural networks is considered. The learning rate is set to be a constant and the momentum factor an adaptive variable. Both the weak and strong convergence results are proved, as well as the convergence rates for the error function and for the weight. Compared to the existing convergence results, our results are more general since we do not require the error function to be quadratic.**

***Index Terms*—Convergence, feedforward neural network, gradient method, momentum.**

I. Introduction

Feedforward neural networks (FNN) have been widely used in applications, and the convergence of the training iteration procedure for FNN by use of the gradient method is discussed in [3]–[5] and [10]. As a simple example, the convergence for two-layer feedforward neural networks is discussed in [5], [6], [10], [11]. To speed up and stabilize the training iteration procedure, a momentum term is often added to the increment formula for the weights [7], [8]. The convergence of the gradient method with momentum is discussed in [1] and [9] under the restriction that the error function is quadratic. A special choice of the momentum term is proposed in [2] without proof of convergence. This momentum term is also used for the two-layer neural network in this paper. (We remark that the learning formula in [2] is a little bit more complicated than ours.) Our contribution in this paper is to prove the convergence of the resulting gradient method with momentum, whereas we do not assume the error function to be quadratic. Some techniques in [10] for gradient methods without momentum has been exploited here in the proof.

We expect to generalize the results in the future to the more general and more important case, i.e., the multilayer BP neural network, which is not straightforward but seems hopeful.

The rest of this paper is organized as follows. In Section II, we introduce the gradient method with momentum. In Section III, we discuss the convergence property of the gradient method with momentum for a two-layer feedforward neural network. We set the learning rate η a constant and the momentum factor τ_k an adaptive variable. Both the

Manuscript received July 27, 2004; revised June 11, 2005. This work was supported in part by the National Natural Science Foundation of China, and the Basic Research Program of the Committee of Science, Technology and Industry of National Defense of China.

N. Zhang is with the Mathematics and Information Science College, Wenzhou University, Wenzhou 325035, P. R. China (e-mail: naiminzhang@yahoo.com.cn).

W. Wu and G. Zheng are with the Department of Applied Mathematics, Dalian University of Technology, Dalian 116024, P. R. China (e-mail: wuweiw@dlut.edu.cn).

Digital Object Identifier 10.1109/TNN.2005.863460